Color Feature Extraction

4

Every picture tells a story, by colors.

4.1 Introduction

Arguably, color is the most important feature of an image. After all, people see this world as colors or the world presents itself to us as colors. However, color is a complex topic and difficult to understand. As a matter of fact, few people are good at painting a picture or image. There are infinite number of colors in this world and colors can be created from different types of palettes. Computers use a trichromatic palette to mix all the colors in this world. That means each color in computers is represented as a three-dimensional vector (c_1, c_2, c_3). These color vectors created a 3D color space. Depending on how each of the trichromatic colors is defined, different color spaces or color models have been created.

The most commonly used color space is the RGB color space, where each of the colors is defined by adding three primary colors in the visible light spectrum (red, green, and blue) with various proportions. Other commonly used color spaces include LUV, HSV/HSL/HSI, YCrCb.

Color spaces are models for the representation of pixel values. To compare and classify color images, however, we need to analyze and understand the *color patterns* in an image. In order to understand the color patterns in an image, we extract color features from the image and compare them with features of other images. Color features are usually based on color statistics computed from an image or regions of an image.

A number of color features have been proposed in the literature including color *histogram*, *color moments* (CM), *color coherence vector* (CCV), color *correlograms*, etc. MPEG-7 also standardizes a number of color features including *dominant color descriptor* (DCD), *color layout descriptor* (CLD), *color structure descriptor* (CSD), and *scalable color descriptor* (SCD).

D. Zhang, *Fundamentals of Image Data Mining*, Texts in Computer Science, https://doi.org/10.1007/978-3-030-17989-2_4

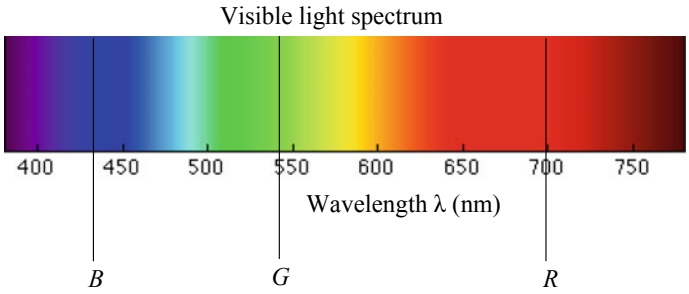

Fig. 4.1 Visible light spectrum and the tristimulus

4.2 Color Space

To process and analyze color images, we need to understand how different color models work, and their applications in image processing and analysis. Color is a complex theory, there are infinite number of colors in this world and colors can be created using a variety of ways. There needs a standard color model so that colors can be reproduced with accuracy and colors produced in different applications by different devices can be translated interchangeably. The first step is to find a way of representing each color numerically and identify the space or gamut of all visible colors.

The building of a standard color model is made possible thanks to the three scientists: *Isaac Newton*, *James Clerk Maxwell,* and *Hermann Grassmann*. *Newton* laid the foundation of our understanding of colors by first splitting a light source into spectral or pure colors (rainbow colors, Fig. 4.1). This lets us to understand that a light is a mixture of pure colors and colors are just reflectance of lights of different wavelengths by objects. *Maxwell* found that by projecting and superimposing the three red, green, and blue monochromatic pictures on the screen, other colors in the scene such as orange, yellow, purples, etc., also showed up, suggesting other colors can be created by mixing red, green, and blue colors. *Grassmann* found that colors are additive. That means any color can be matched by a linear combination of three other colors (primaries), provided that none of those three can be matched by a combination of the other two; and a mixture of any two colors can be matched by linearly adding together their primary components.

4.2.1 CIE XYZ, *xyY* Color Spaces

Modern color models are all based on XYZ color space created by CIE (International Commission on Illumination) in 1931. The CIE XYZ color space was created using Maxwell's *tristimulus* theory, which is based on the theory of *trichromatic* color vision found by Young and Helmholtz [1], who discovered that human vision consists of three types of cones, which are sensitive or responsive to three narrow

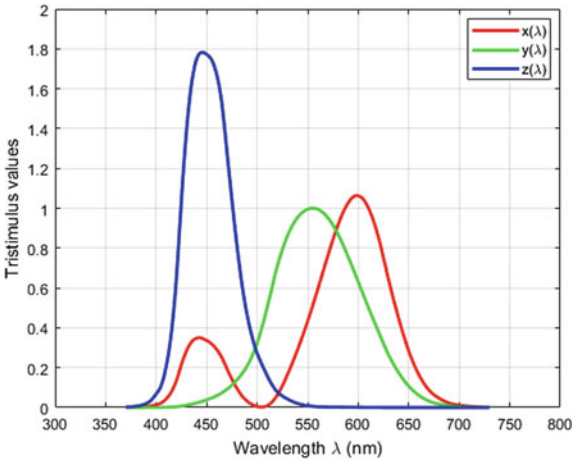

Fig. 4.2 CIEXYZ color matching functions of human vision

bands of visible lights. Figure 4.2 shows the three color matching functions which indicate human eyes' response to visible colors. It can be observed that three functions peak at around 600 nm, 550 nm, and 450 nm, respectively. It demonstrates that human vision is most sensitive to three bands of lights, which are perceived as red, green, and blue.

Based on this discovery, CIE uses three *primary colors* to match out all the *spectral colors*, i.e., pure colors or colors with a single wavelength (Fig. 4.1) [2]. The three primary colors are all pure colors, they are R (700 nm), G (546.1 nm), and B (435.8 nm), respectively, which are shown on the visible color spectrum in Fig. 4.1. The choice of the three particular primaries was due to practical reason at that time. The primaries G (546.1 nm) and B (435.8 nm) were chosen because they could easily be reproduced using mercury excitation lines. The 700 nm primary color was chosen is because the hue near that wavelength is homogenous and nearly constant, therefore, slight inaccuracy in production of the wavelength of this spectral primary would introduce no error at all.

The three primary stimulus are projected on a screen with relative power and are mixed/added by various proportions to match out each of the *spectral colors* in the visible color spectrum using Grassmann's laws. Each color can now be represented as a three-value tuple (R, G, B). The R, G, B values are then normalized using the formulas in (4.1) to remove the intensity from the color representation. The normalised r, g, b values are purely chromatic values. This creates three color matching functions (CMF) $r(\lambda)$, $g(\lambda)$, and $b(\lambda)$, where λ is the wavelength. The color space created based on the three CMFs is called CIE RGB color space.

$$\left.\begin{aligned} r &= \frac{R}{R+G+B} \\ g &= \frac{G}{R+G+B} \\ b &= \frac{B}{R+G+B} \end{aligned}\right\} \tag{4.1}$$

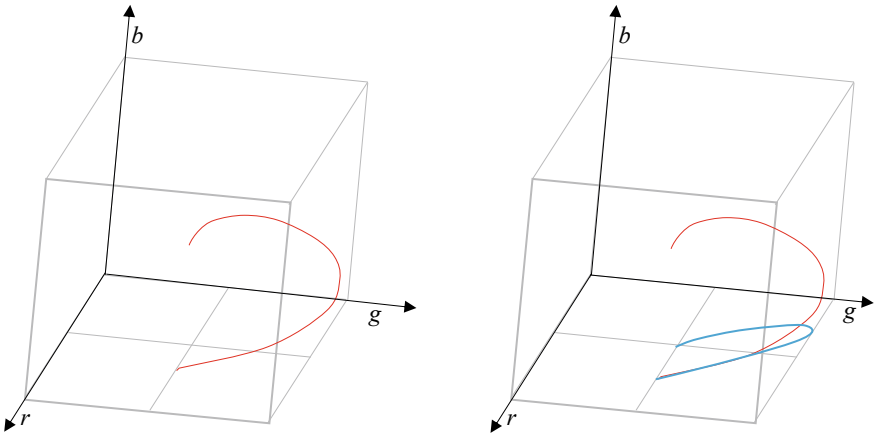

Fig. 4.3 Color matching function of spectral colors. Left: *rgb* curve of spectral colors; Right: projection of the *rgb* curve onto the 2D *rg* plane (cyan)

Now if we plot the (r, g, b) coordinates of all the spectral colors in 3D space, it forms a curve (Fig. 4.3 Left) [3]. It is easy to see from (4.1) that $r + g + b = 1$ or $b = 1 - r - g$, which means b is a dependent function of r and g, so there is no need to keep the information b. Therefore, by projecting the *rgb* curve into the 2D *rg* plane, we get the horseshoe-shaped 2D *rg* curve which is shown as cyan color in Fig. 4.3 Right.

This *rg* curve is then transformed to the CIE *xy* curve by aligning the $g(\lambda)$ with the CIE luminosity function $V(\lambda)$ and removing the negative values in $r(\lambda)$. The color space created based on the *xy* curve is called CIE XYZ color space.

The colors on the 2D *xy* curve are all *spectral colors*, to obtain *nonspectral colors*, i.e., mixed colors or colors with multiple wavelengths, we draw a straight line between any two points on the *xy* curve. Then each point on the straight line represents a nonspectral color mixed by the two colors at both ends of the line according to Grassmann's second law. For example, by connecting the two primary colors *R* (700 nm) and *B* (435.8 nm) on the *xy* curve, purple colors are created. By this way, all possible *nonspectral colors* can be created. In practice, a white color or white point is first defined, such as D65 which represents the midday Sunlight, and lines are drawn from the white point to each of the spectral colors on the *xy* curve. Then colors of different purity are created by mixing the white color with each of the pure colors on the curve. Figure 4.4 shows the color gamut of CIE XYZ color space or CIE *xy* gamut [2]. The color gamut created in this way is a *hue* and *saturation* gamut.

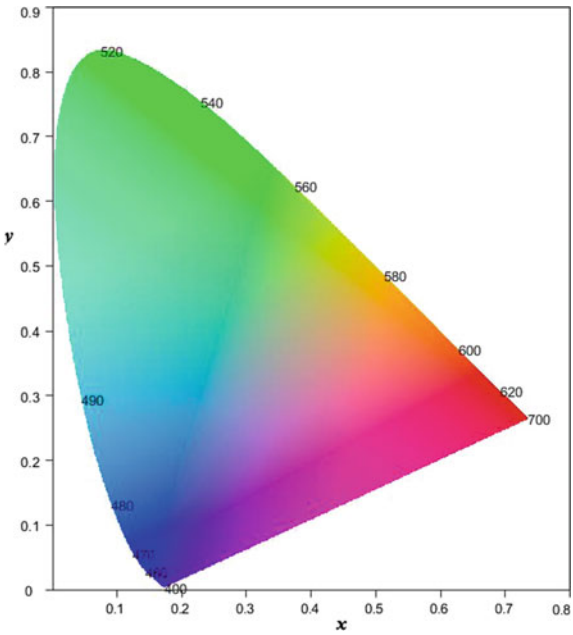

Fig. 4.4 CIE *xy* color gamut

CIE *xy* gamut is a chromaticity domain, it does not specify luminance or brightness of colors. To create object colors, the luminance/brightness must be given as the third dimension, named as *Y*. Therefore, CIE *xyY* color space is created and is called the *object color space*, where *x* and *y* are chromaticity values, and *Y* is the luminance value.

CIE XYZ color space is a cornerstone for modern color modeling. The significance of CIE XYZ color space can be summarized in the following:

- Provides a color gamut with all possible colors,
- Specifies each color with a three-value tuple or a 3D vector (*x*, *y*, *z*),
- Provides a reference for all other color models,
- It is a device-independent color space.

4.2.2 RGB Color Space

Digital images are generated using RGB colors. RGB colors are device dependent, which means that each type of digital devices typically uses a different set of RGB primaries to generate colors. For example, computers use a standard RGB color model or sRGB, which is based on the following three primaries chosen from the CIE XYZ color gamut.

Chromaticity	x	y	Y
R	0.6400	0.3300	0.2126
G	0.3000	0.6000	0.7125
B	0.1500	0.0600	0.3290
W(hite)	0.3127	0.3290	1.0000

The three primaries of sRGB color model are shown in Fig. 4.5 [4]. The gamut of sRGB color space is a triangle inside the CIE XYZ gamut.

The RGB color gamut can be regarded as a color palette. All possible colors created by the RGB palette can be visualized in a 3D cube, called RGB color space as shown in Fig. 4.6. The colors in the RGB color cube are usually quantized for the convenience of viewing. Given the pixel values (r, g, b) from a color image, a color c can be defined in the RGB color space and reproduced by mixing the three primaries using $c = rR + gG + bB$.

It is clear that the gamut of any RGB color model is a triangle inside the CIE XYZ color gamut, consequently, a RGB color space cannot represent all visible colors. In case of a color out of the RGB color gamut, an approximation has to be made. For a color C out of the RGB gamut, the approximation of C is given by the color C_A at the intersection of the RGB triangle and line CW which is the connection between color C and white point W. The approximation is usually acceptable because C_A is just a desaturated color from C.

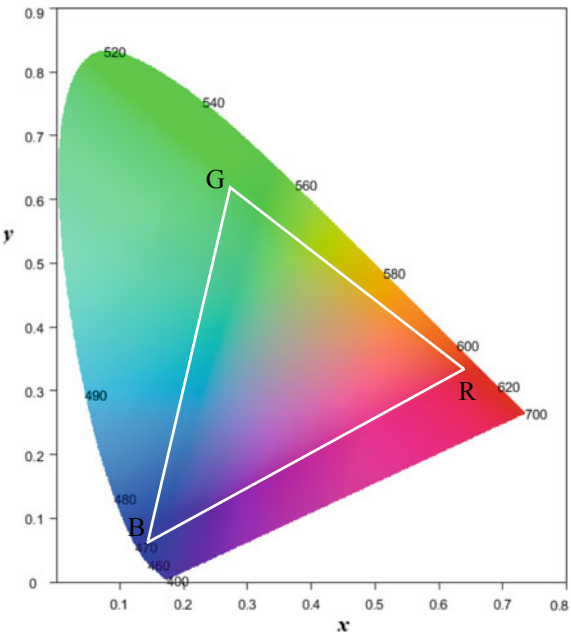

Fig. 4.5 The sRGB triangle gamut shown inside the CIE XYZ gamut

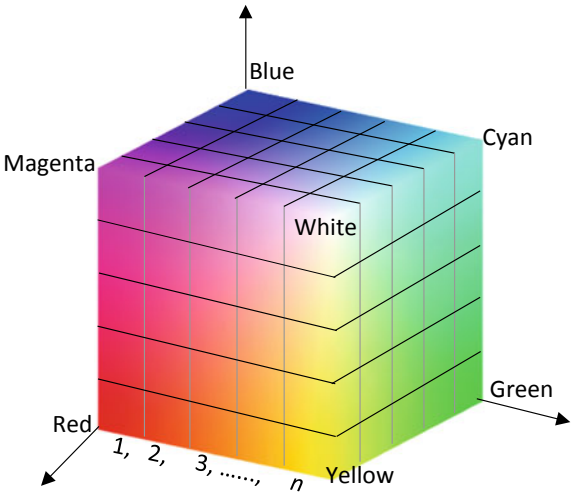

Fig. 4.6 RGB color space in 3D

RGB module is useful for color display and printing, however, it is not desirable for image processing and analysis. This is because the three channels are dependent on each other and there is a high correlation between the three channels, which means, change any one of the color channels will change the other two color channels. Furthermore, RGB color space is not perceptually uniform, meaning that the same amount of numerical change in color values does not correspond to about the same amount of visually perceived change. This leads to color spaces with separation of luminance from chromaticity and color spaces with uniform color distance.

4.2.3 HSV, HSL and HSI Color Spaces

RGB color model is efficient because it just uses three primaries to create all required colors. However, the RGB color model is not intuitive because it does not conform to how human beings understand and make colors. For example, artists and painters do not use RGB mixture to make colors, instead, they use pigments to mix with either white or black or both (gray) to make required colors. The pigments are equivalent to pure colors or spectral colors, when they are mixed with white or black, lighter or darker colors are created; when they are mixed with gray, colors with different purity or saturation are created. Figure 4.7a shows how different tints, shades, and tones of reddish colors are created by artists.

The way artists and painters making colors is the idea behind the HSV color model. It demonstrates that a color can be specified by three components/properties: Hue, Saturation, and Value or (H, S, V), where Hue is a pure color and Value = Brightness. The *Hue* tells what color it is, it is determined by the dominant wavelength on the visible color spectrum (Fig. 4.1). The *Saturation* tells how much or how colorful is the color, the more saturated the color, the more vibrant or vivid

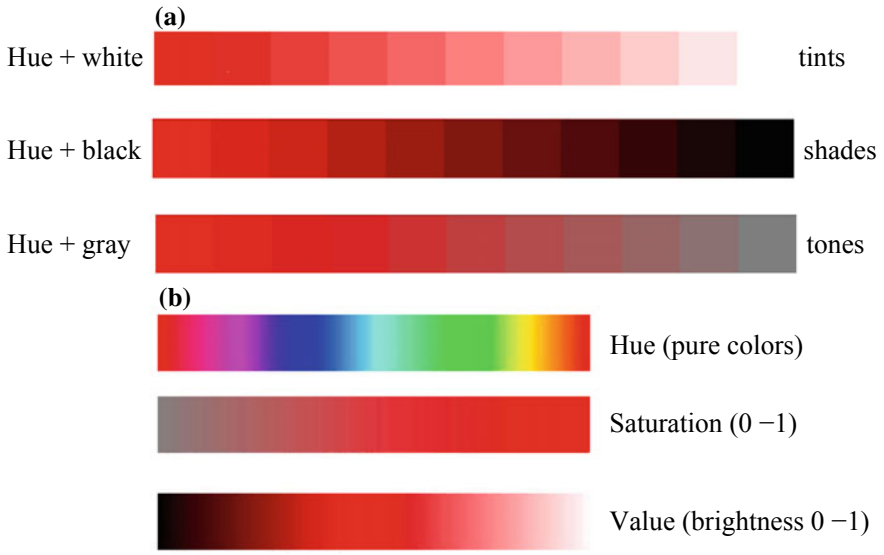

Fig. 4.7 **a** Artists' way of making reddish colors. **b** Components of HSV

the color is. The *Value* tells how bright or dark is the color, colors become disappearing when they are too dark or too bright. Therefore, a color specified by the (h, s, v) values makes a lot more sense than that given by the (r, g, b) values. Multimedia editors and image processing software nowadays all provide intuitive HSV color picker simply because users have better chance to make the desired colors using HSV model than using other color models.

Figure 4.7b demonstrate the HSV color making using red colors as an example. The first bar shows all the pure colors or hues; the second bar shows red colors with a different purity or saturation (but with the same brightness); the third bar shows red colors with different brightness.

To create the HSV color model, pure colors (spectral colors) are first collected and put on a circle or a ring (Fig. 4.8 Left), colors with different saturation are created along the radii of the circle to create a hue–saturation disk/wheel (Fig. 4.8 Right). Hue–saturation disks with different brightness are then generated and stacked on top of each other to make a color cylinder which is the HSV color space, shown in Fig. 4.9a. For HSV, the most saturated colors are on the top of the cylinder and the top of the cylinder has a V value of 1. For HSL, the most saturated colors are in the middle of the cylinder and the top of the cylinder is the white color (L = 1), this is shown in Fig. 4.9b.

It is observed from the Value strip of Fig. 4.7b that as colors become darker or brighter, they become less colored, consequently, as shown in Fig. 4.9a, b, colors on the HSV and HSL cylinders become more and more redundant as they go down the cylinders and as they go up the HSL cylinder. Therefore, the actual HSV color space is often shown as a single cone (Fig. 4.9c) while HSL color spaces is often

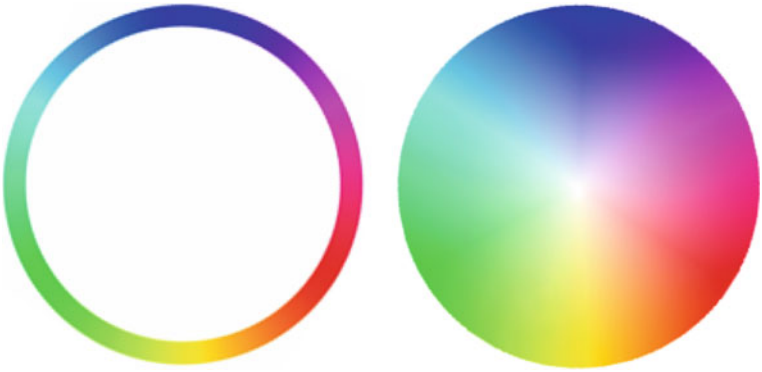

Fig. 4.8 Hue and saturation. Left: pure colors on a ring; Right: hue-saturation wheel

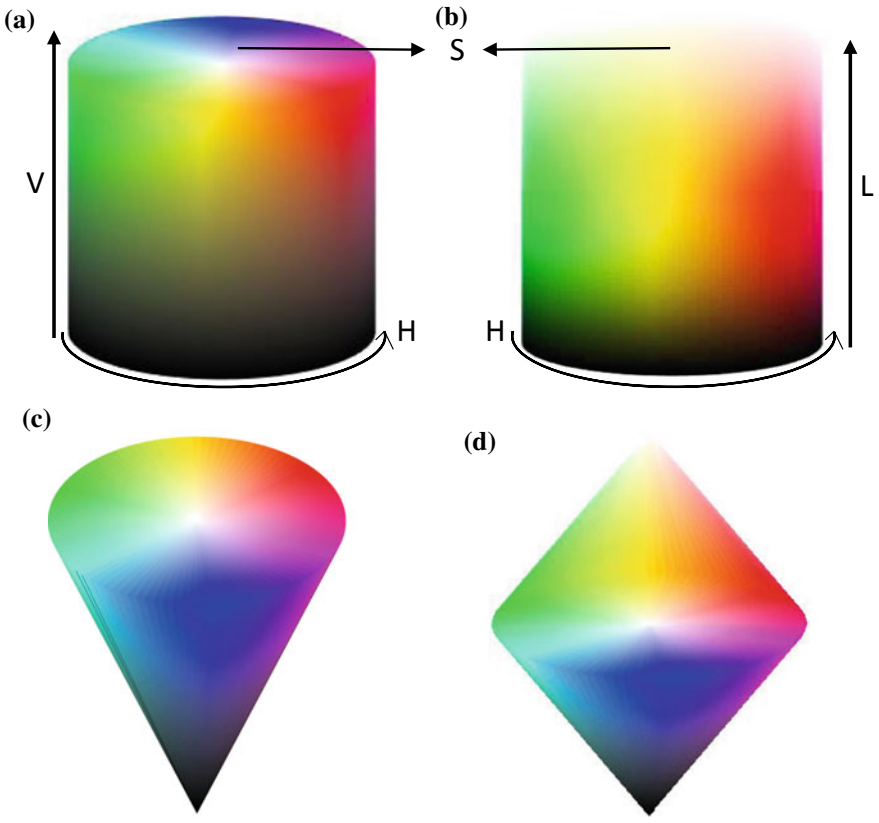

Fig. 4.9 HSV and HSL color spaces. **a** HSV cylinder; **b** HSL cylinder; **c** HSV cone; **d** HSL double cone

shown as a double cone (Fig. 4.9d). Because every slice of the HSV/HSL cone is colorful, the radiuses of the cone are called *chroma* instead of *saturation*.

Digital images are represented using RGB colors. To derive HSV, HSL and HSI color values, RGB values are normalized into 0–1. Given a (R, G, B) color, the H, S, V values are computed using the following guidelines.

- The Hue values are organized into 0°–360° around a pure color circle, with red color at 0°/360°, green at 120°, and blue at 240°;
- To determine the H value of a (R, G, B) color, the maximum of the three values is used to determine the *dominant hue* on the pure color circle and the difference of the other two values tells what side is the RGB color located at the *dominant hue*;
- The Value/Intensity/Lightness of the (R, G, B) color is determined by either the maximum of the three RGB values or the average or in between;
- The Saturation of a color is determined by how far the color is from the pure color circle (Hue) which has a color saturation of 1. Saturation is given in percentage, e.g., 40%.

Let

$$\begin{aligned} M &= max(R, G, B) \\ m &= min(R, G, B) \\ C &= M - m \end{aligned}$$

The maximum of the three RGB channels M dominants the *hue* and *brightness* of a color. The C value, called *chroma*, is proportional to *saturation* of a color. With these in mind, HSV values can be computed using the following formulas:

$$H = \begin{cases} 0 & if\ M = m \\ 60° \times \frac{G-B}{C} + 0°, & if\ M = R\ and\ G \geq B \\ 60° \times \frac{G-B}{C} + 360°, & if\ M = R\ and\ G < B \\ 60° \times \frac{B-R}{C} + 120°, & if\ M = G \\ 60° \times \frac{R-G}{C} + 240°, & if\ M = B \end{cases} \tag{4.2}$$

$$S = \begin{cases} 0 & if\ M = 0 \\ \frac{C}{M} = 1 - \frac{m}{M}, & otherwise \end{cases} \tag{4.3}$$

$$V = M \tag{4.4}$$

For *HSL* model:
H is the same as (4.2)

$$L = \frac{1}{2}(M + m) \tag{4.5}$$

$$S = \begin{cases} 0 & if\ M = m \\ \frac{C}{2L}, & if\ L \leq \frac{1}{2} \\ \frac{C}{2-2L}, & if\ L > \frac{1}{2} \end{cases} \tag{4.6}$$

For *HSI* model:
H is the same as (4.2)

$$I = \frac{1}{3}(R + G + B) \tag{4.7}$$

$$S = \begin{cases} 0 & if\ I = 0 \\ \frac{I-m}{I} = 1 - \frac{m}{I}, & otherwise \end{cases} \tag{4.8}$$

HSV values can also be derived using the following formulas:

$$H = \arctan\left(\frac{\beta}{\alpha}\right) \tag{4.9}$$

$$S = \sqrt{\alpha^2 + \beta^2} \tag{4.10}$$

$$V = max(R, G, B) \tag{4.11}$$

where

$$\alpha = R - \frac{1}{2}(G + B), \quad \beta = \frac{\sqrt{3}}{2}(G - B) \tag{4.12}$$

The H and S components are invariant to lighting variations or intensity changes. Intensity changes are only reflected in the V component, which can be corrected by a linear scaling.

Figure 4.10 shows a color image and its H, S, V channels. In the H image, white and black are starting and arrival points on the color wheel, they represent Red color and Yellow color. Gray intermediate levels are corresponding to intermediate hues on the wheel. Both the S and V channels are in the 0–1 range. For S channel, white is pure color and black is minimum saturation. For V channel, white is very bright and black otherwise.

It can be observed from the S and V channels that both the yellow plants at the bottom left of the image are highly saturated and also very bright, while the red leaf tree is highly saturated but with moderate brightness. Notice the shadow at the bottom left of the color image has no to little effect on the H and S channels, it only affects the V channel.

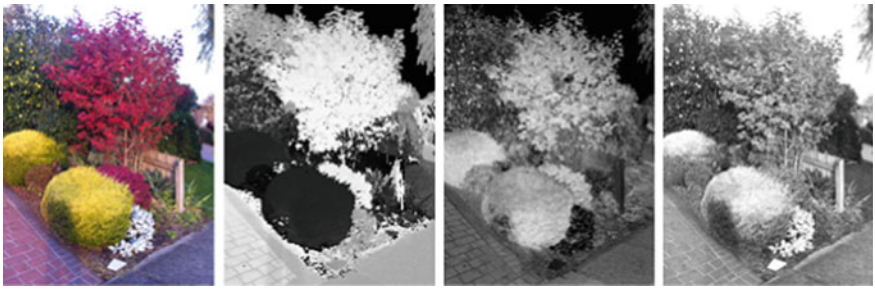

Fig. 4.10 A color image on the left and its H, S, V channels on the right columns

4.2.4 CIE LUV Color Space

The CIE XYZ color space is nonuniform in terms of color differences, so is RGB color space. Nonuniform means that the calculated difference between the two colors is not proportional to their perceived color difference. This phenomenon is shown as MacAdam ellipses in the CIE *xy* gamut in Fig. 4.11. Each ellipse shown in the figure represents colors within the just-noticeable-difference (JND) threshold. In other words, colors within each ellipse are perceivably the same. As can be seen, the sizes of the ellipses in different areas of the gamut vary significantly. This implicates that colors within certain distance may be perceived as the same color in one area but as different colors in another area. This causes confusions for many

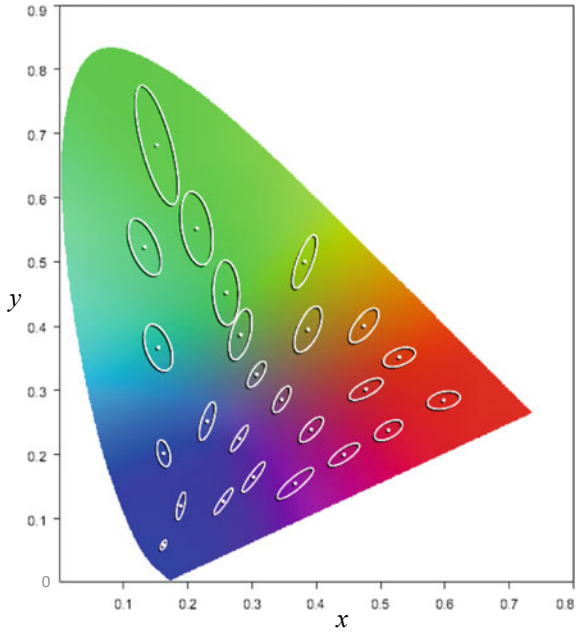

Fig. 4.11 MacAdam ellipses (magnified 10 times) on the CIE *xy* gamut

color applications including displaying, image processing, and image analysis, which rely on computing color differences.

To overcome the nonuniform color spread problem, CIE Luv (1960) and CIE Lu′v′ (1976) color spaces have been created. Both Luv and Lu′v′ spaces are transformed from CIE XYZ space, and the two color spaces are only different at the v component. The idea is to stretch or squeeze the CIE xy gamut at certain directions so that the MacAdam ellipses are made to equal size.

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 & if\ \frac{Y}{Y_n} > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right) & if\ \frac{Y}{Y_n} \le 0.008856 \end{cases} \tag{4.13}$$

$$u = \frac{4X}{X + 15Y + 3Z} = \frac{4x}{3 - 2x + 12y} \tag{4.14}$$

$$v = \frac{6Y}{X + 15Y + 3Z} = \frac{6y}{3 - 2x + 12y} \tag{4.15}$$

$$u' = u, \quad v' = 1.5\,v \tag{4.16}$$

where Y_n is the luminance of the white point and

$$\left.\begin{array}{l} x = \frac{X}{X+Y+Z} \\ y = \frac{Y}{X+Y+Z} \end{array}\right\} \tag{4.17}$$

L^* scales from 0 to 100 due to the relative luminance (Y/Y_n) scales from 0 to 1. The cubic root function of L^* is nonlinear and is intended to mimic the *logarithmic response* of human eyes to lightness. The transformed uv and $u'v'$ gamuts marked with the MacAdam ellipses are shown in (4.12). It can be seen that the differences between the sizes of the ellipses are considerably reduced (refer to Fig. 4.11) (Fig. 4.12).

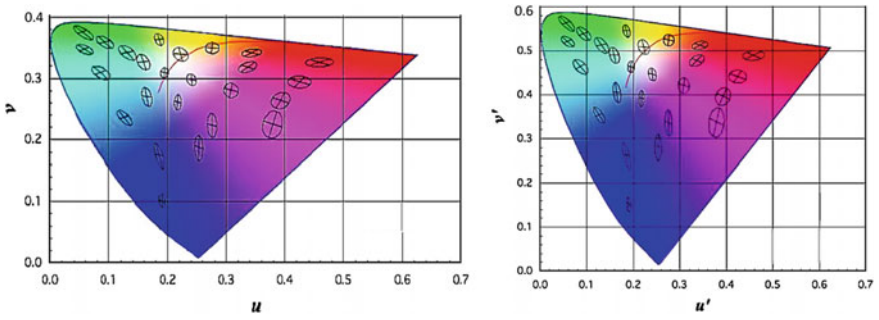

Fig. 4.12 CIE Luv and Lu′v′. Left: MacAdam ellipses on CIE uv gamut; Right: MacAdam ellipses on CIE $u'v'$ gamut

The nonuniformity can be further reduced by using the u^*v^* chromaticity:

$$u^* = 13L^*\left(u' - u'_n\right) \tag{4.18}$$

$$v^* = 13L^*\left(v' - v'_n\right) \tag{4.19}$$

where (u'_n, v'_n) are the coordinates of the white point on the (u', v') gamut.

To derive LUV from RGB color space, RGB values are first transformed into XYZ values using the following matrix, where RGB values have been normalized to 0–1.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1804 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9503 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.20}$$

Although CIE LUV color spaces are close to uniform color spaces, the source RGB primaries are assumed to be known, so that a specific transform matrix of (4.20) can be used. This can be an issue for image applications, because the source (device) RGB primaries are usually unknown.

4.2.5 Y′CbCr Color Space

Both HSV and LUV color spaces are based on the same idea, i.e., the separation of luminance from chromaticity. This idea has been found ideal and desirable for most of the color applications including image processing and feature extraction. Y′CbCr is another such kind of color space, which is often used for image compression and representation. The transformation from RGB space to Y′CbCr space is given by the following equation:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.21}$$

where Y′ is the luminance, Cb is the blue component, and Cr is the red component. Both RGB values and Y′CbCr values are in the range of [0, 255].

By separating the luminance Y′ from the chromaticity Cb and Cr, most of the image information has been concentrated onto Y′. This is ideal for image communication and representation, because each channel can now be treated independently instead of treating all the three channels equally as in the RGB space. For example, more importance can be given to Y′ as in many situations. This makes the communication more efficient (e.g., fewer bits for color channels) and representation more compact.

Figure 4.13 shows an example of Y′CbCr channels from a flower image. It can be seen that the Y′ channel is basically the gray level version of the original image,

Fig. 4.13 The flower image on the leftmost and its Y′, Cb, and Cr channels on the right

while the Cb and Cr channels are just for the color information, both Cb and Cr channels contain very little information compared with the Y′ channel. In the TV broadcast, the Y′ channel can be sent out independently to be compatible with the old noncolor TVs.

4.3 Image Clustering and Segmentation

Digital images are complex data. Unlike textual documents which are made of words from a dictionary of a small vocabulary, there is no visual dictionary or vocabulary for images. Each image consists of thousands to millions of pixels which represent color values, and the possibilities of the pixel colors are almost infinite. Therefore, the first step to analyze an image is usually to group the image pixels into a small number of regions or objects so that further analysis can be carried out, this is called *image clustering or segmentation*. There are many segmentation and clustering algorithms in the literature, in the next, we discuss two widely used algorithms in image feature extraction.

4.3.1 *K*-Means Clustering

One of the simplest segmentation methods is the *K-means* clustering. *K*-means clustering attempts to divide a dataset into *K* clusters with each data point belonging to the cluster with the closest mean, which serves as the centroid of the cluster. The *K*-means clustering algorithm is given as follows:

K-means (*K*) {
Input: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, a set of data points
Output: $C = \{c_1, c_2, \ldots, c_K\}$, a set of clusters

1. Randomly select *K* cluster centers or seeds;
2. Calculate the distance between each data point and all the *K* cluster means $\mathbf{m}_k$;
3. Assign the data point to the cluster with the nearest cluster mean;

4. Recalculate the new means for each cluster;
5. Repeat from step 2 until no data point needs to be reassigned.

}

K-means clustering algorithm aims at minimizing an objective function known as the *sum of squared error* or SSE function, which is given by

$$J(C) = \sum_{k=1}^{K} J_k \tag{4.22}$$

and J_k is the SSE of the kth cluster:

$$J_k = \sum_{i=1}^{|c_k|} (\|\mathbf{x}_i - \mathbf{m}_k\|)^2 \tag{4.23}$$

where $\mathbf{m}_k$ is the mean of cluster c_k and $|\, c_k \,|$ is the number of instances in cluster c_k.

K-means is one of the most commonly used clustering algorithms in image processing and analysis. It is especially useful for many color-based clustering such as *color quantization*, which aims to group similar colors and reduce the number of colors in an image. The key issue with a *K*-means clustering algorithm is the parameter *K*. The performance of the clustering depends on a good guess of *K*, however, there is no easy solution. This leads to other more sophisticated algorithms to improve the method.

4.3.2 JSEG Segmentation

JSEG method is based on the belief or assumption that color regions and textures agree with each other in an image, which means that a region with similar colors also has a similar texture. Based on this idea, the method attempts to find an agreement between the two types of features. The procedure of JSEG is summarized as follows [5]:

- **Color quantization**. At first, pixel colors of the image are quantized into a number of classes using a clustering algorithm such as *K*-means clustering.
- **Color map**. Pixels in the image are then replaced with the color class labels, such as 1, 2, 3, …. A class map is then formed and region growing is followed on the class map (Fig. 4.14b).
- **J-image**. The key to JSEG method is the computing of a *J-image,* which is computed by moving a local window through each pixel and calculating the SSE over the window (4.23). The SSE is related to the variance over a local neighborhood, neighborhoods with relatively uniform colors (or little to no texture) tend to have small *J* values while neighborhoods with high *J* values correspond to region boundaries or edges. The window size determines the sharpness of the *J*-

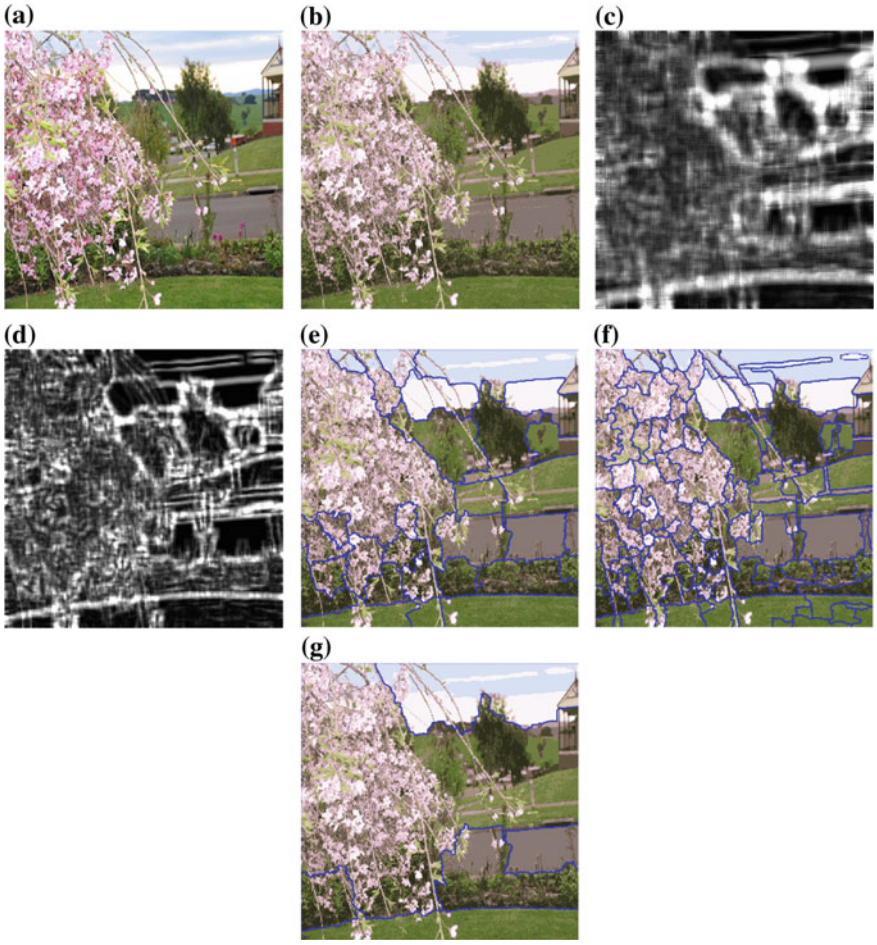

Fig. 4.14 An image segmentation using JSEG. **a** An original color image; **b** result of color quantization with 13 colors; **c** *J*-image at scale 3; **d** *J*-image at scale 2; **e** segmentation result at scale 3; **f** segmentation result at scale 2; **g** final result of segmentation after merging

image and the size of the regions that can be detected. The *J*-image computed using larger local window is more blurred than that computed using a smaller window (Fig. 4.14c, d).

- **Region growing**. Based on the *J*-image, a region growing method is carried out starting from areas with the lowest *J* values. After each region growing, the total *J* values of each region k (J_k of (4.23)) and the average *J* values ($\bar{J}$) of all the J_k are computed. The region growing is then repeated using a *J*-image with smaller scale until the $\bar{J}$ value stops decreasing (Fig. 4.14e).

- **Merging**. The region growing can result in over segmentation due to texture variations (Fig. 4.14f). Therefore, a merging process is followed by merging *J*-segmented regions with similar colors (Fig. 4.14g).

Due to the use of both color and texture features and a merging process, JSEG gives a less fragmented segmentation than *K*-means clustering. However, the performance of JSEG segmentation depends on several parameters such as the numbers of quantized colors, the seed selection threshold during the region growing, and the threshold of color similarity during the region merging. The computation is also very expensive, the segmentation of a 512 × 512 image can take about 4 min on a PC.

4.4 Color Feature Extraction

4.4.1 Color Histogram

The simplest feature of a color image is its histogram, which describes the color distribution within an image. To create a histogram for an image, a number of bins (*N*) are first created, each bin represents a group of similar colors. Each pixel in the image is then examined and put into a bin with similar colors to the pixel. After all pixels in the image are checked, the pixels in each bin are counted and each bin is represented as a value which is the number of pixels in the bin. A bar chart consists of all the *N* bin values is created and it's called a color histogram, which is a sequence of (*c*, *n*) pairs shown in a graph, where *c* is the color of the bin and *n* is the number of pixels in the bin. Figure 4.15 shows the three color histograms of the R, G, B channels of the Lena image in one graph.

In terms of histogram feature extraction, there are three ways to create a histogram for a color image: *component histogram*, *indexed color histogram*, and *dominant color histogram*.

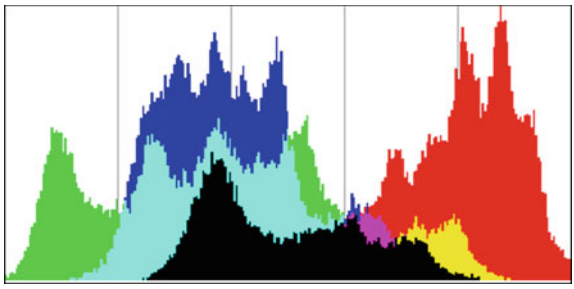

Fig. 4.15 RGB histograms of the Lena color image. Non-RGB colors are the areas of overlap between the R, G, B channels. Each channel is quantized into 256 colors or bins, which are on the horizontal axis, vertical axis shows the number of pixels in each bin or color

4.4.1.1 Component Histogram

The first way of color histogram is to split a color image into individual R, G, B channels (Fig. 4.16 top row), each individual channel is equivalent to a gray level image (Fig. 4.16 bottom row). A contrast with Fig. 4.13 tells that there is a lot of redundancy or correlation between R, G, and B channels. A histogram is first created for each individual channel. The three individual channel histograms can then be concatenated into a single histogram. If each individual color channel is represented by l, m, and n bins, respectively, the final histogram will have $N = l + m + n$ bins.

For example, for Lena image in Fig. 4.15, each R, G, B channel is represented by 8 bits and a total of 256 colors/bins. By concatenating the three histograms, the final histogram would have $N = 3 \times 256 = 768$ bins. This is, however, too long for image representation, therefore, the colors of each individual R, G, B channel is usually quantized to reduce the number of colors.

To quantize the color channels, colors in each channel are divided into equal intervals and each interval is used as a bin. For example, to create a 4-bin histogram for R channel, the 256 colors in R channel are divided into the 4 intervals: (0, 63), (64, 127), (128, 191), (191, 255). Figure 4.17 shows a 216-bin histogram by quantizing each of the R, G, B histograms in Fig. 4.15 into 72 bins. This is less than one-third of the length of the histogram without color quantization.

Fig. 4.16 RGB channels of a color image. Top row: R, G, B channels of the flower image; Bottom row: corresponding gray level images of the R, G, B channels at the top row

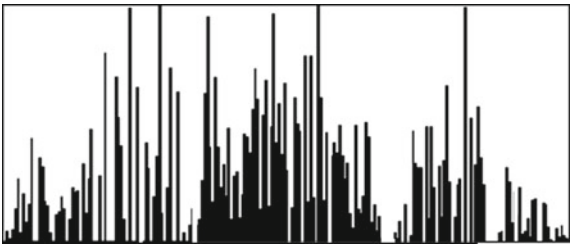

Fig. 4.17 Concatenation of histograms of individual R, G, B channels into a single histogram (216 bins, 72 bins for each channel)

4.4.1.2 Indexed Color Histogram

Another way to create a color histogram is to quantize the RGB color space (instead of each color plane) into N colors and use the N colors as bins to create a color histogram. This is equivalent to indexed colors and the N colors are equivalent to a global color palette, i.e., a palette representing all image colors or a palette for all the images in the world.

To quantize the RGB color space, the R, G, B planes are divided into l, m, n intervals, respectively, using the same way as in the first method, the RGB color space (a big cube) is then divided into $N = l \times m \times n$ small color cubes (Fig. 4.6). Each small cube represents a group of similar colors and is used as a histogram bin.

To create a histogram for a color image using the indexed colors, each pixel in the image is examined and put into a bin with similar colors to the pixel. A histogram is then created by counting the number of pixels in each of the bins. Figure 4.18a shows the flower image with 216 quantized or indexed colors and the 216-bin histogram for the quantized color image is shown in Fig. 4.18b.

4.4.1.3 Dominant Color Histogram

A histogram created from a global palette (a single fixed palette for all images) is usually sparse (Fig. 4.18b), this is because when a global palette is used, most of the colors in an image are often missing from the color representation, this effect is shown up in both the quantized image and the color histogram (Fig. 4.18a, b). In practice, an adaptive or native palette created from the image itself can be used, a histogram created from *adaptive indexed colors* is essentially a *dominant color histogram*. Figure 4.18c, d shows the flower image quantized with an adaptive palette of 216 colors and its histogram. Dominant colors can be obtained by either using a histogram thresholding or using a K-means clustering.

A histogram is invariant to translation and rotation changes, scale invariance can be achieved by normalizing each bin value with the total number of pixels in the image. The key issue with a histogram is the difficulty to determine the number of bins. If the number of bins is too small, the colors in a bin can vary so much that it causes too many confusions during the matching. On the other hand, if the number of bins is too large, it causes overfitting, means there are too few pixels in a bin, this too can cause confusions during the matching. In practice, there is always a

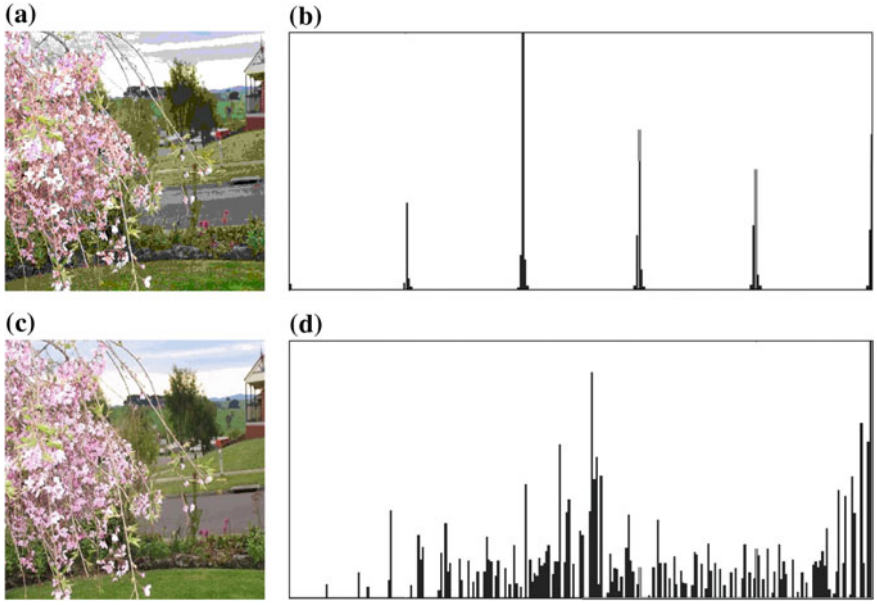

Fig. 4.18 Indexed color histograms of a color image. **a** The flower image quantized with a global palette of 216 indexed colors (notice the visible distortion on the sky and grass); **b** indexed color histogram of (**a**) (216 bins); **c** the flower image quantized with an adaptive palette of 216 indexed colors; **d** dominant color histogram of (**c**)

compromise on the number of bins. Regardless, features based on color histograms usually have very high dimensions, e.g., 512 and 1024 dimensions are common.

Another issue with the histogram method is that a color histogram does not tell pixels' spatial information. Therefore, visually different images can have similar color histograms. This is undesirable for image representation. A number of other color feature extraction methods have been designed to address these two issues, they are discussed in the next.

4.4.2 Color Structure Descriptor

One of the key drawback of a histogram is the absence of spatial information of pixel colors in an image. The spatial information of pixel colors tells the patterns of colors, or how colors are spread out inside an image. Without the spatial information of pixel colors, perceptually different images can have the same histogram and this can lead to incorrect image retrieval or classification. For example, Fig. 4.19 shows two perceptually different binary images with the same size and the same number of red pixels. The image on the left is visually structured and the red color is perceived as a regular shape, in contrast, the image on the right is visually

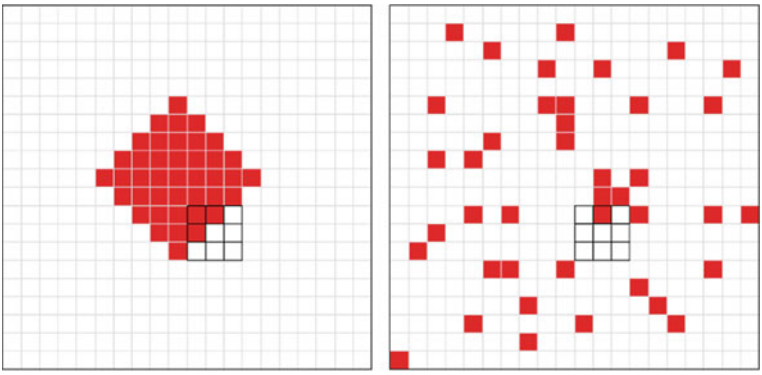

Fig. 4.19 Computation of color structure descriptor. Left: a color image with a 3 × 3 structure moving through the image. The structure captures the red color 56 times; Right: a color image with a 3 × 3 structure moving through the image. The structure captures the red color 218 times

unstructured and the red color is perceived as being cluttered. However, the two images have exactly the same color histogram.

One solution to detect color patterns inside an image is to use a structure element or window as the color picker instead of the pixel color picker/counter used in the ordinary histogram computation. When the window moves throughout an image, only the *colors* (e.g., white, red, gray, brown, etc.) inside the window are counted instead of counting the *pixels* of each color inside the window. The histogram created in this way is called color structure (CS) histogram. A CS histogram has a *multiplying* effect on the counting of isolated or scattered colors, the larger the structure window is used, the more the counting is multiplied. While the CS histogram only has a mild over-counting of grouped or clumped colors.

For example, in Fig. 4.19, both images have 41 red pixels, however, by moving a 3 × 3 structure window throughout the images, the red color in the right image are counted for 218 times, while the red color in the left image is only counted for 56 times. The large difference between the two figures accurately reflects the sharp difference between the two red patterns.

To create a CS histogram for an image, the image is first converted to HSV color space and is quantized into a smaller number of colors, e.g., 256, 128, 64, or 32 colors. A structuring element (e.g., square) is then moved throughout the image. Bin i of the histogram records how many times the structuring element captures at least one pixel with color i. If the window is of size 1 pixel, the CS histogram is just an ordinary histogram. In this sense, the ordinary histogram is just a special case of a CS histogram.

For example, the left-hand side of Fig. 4.20 shows a color image with 5 colors and the 4 × 4 window (black) capturing three types of colors: blue, green and brown [6]. The right-hand side shows how the CS histogram accumulates the three colors (green, blue. and brown) captured by the window into corresponding

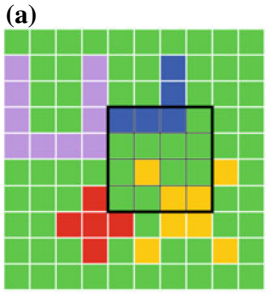

Fig. 4.20 Accumulation of color structure descriptor. **a** A 5-color image and a 4 × 4 structuring element. **b** The accumulation of color structure histogram at a particular position of the structuring element in the image

histogram bins (1, 2, and 4). The CS histogram is then normalized with a total number of counts of the histogram, and the normalized CS histogram is the CSD.

It can be expected that the CSD is more robust than an ordinary histogram, because it captures the information about local spatial structure as well as the color distribution of an image. The spatial information makes the CS histogram sensitive to certain color patterns to which an ordinary histogram is blind.

Furthermore, a CSD uses a window of size greater than 1 pixel, it is less susceptible to noise. However, the performance of CSD depends on the size and structure of the window. Scale invariance can only be achieved by varying the size of the structure element and doing the best match between two images. Rotation invariance can be achieved by using a circular element instead of a squared one.

4.4.3 Dominant Color Descriptor

It is understood that an image is visually interpreted based on a few dominant colors. Those other colors are either noise or just for details, they are not important and can be ignored. Therefore, a dominant color histogram will better describe an image than a common histogram. The *Dominant Color Descriptor* (DCD) is just based on this idea, it's a variation of a common histogram.

To derive a DCD, a histogram h of all colors (without quantization) in an input image I is first created. A thresholding is then applied to h to eliminate those bins whose values are less than a threshold τ. The remaining n colors are called dominant colors. Each dominant color is represented as $(\mathbf{c}_i, p_i)$, $i = 1, 2, \ldots, n$, where $\mathbf{c}_i$ is a 3D color vector and p_i is the percentage of pixels in the image having color $\mathbf{c}_i$. A DCD is just the n dominant colors in a sequence:

$$\mathrm{DCD} = \{(\mathbf{c}_i, p_i), i = 1, 2, \ldots, n\} \tag{4.24}$$

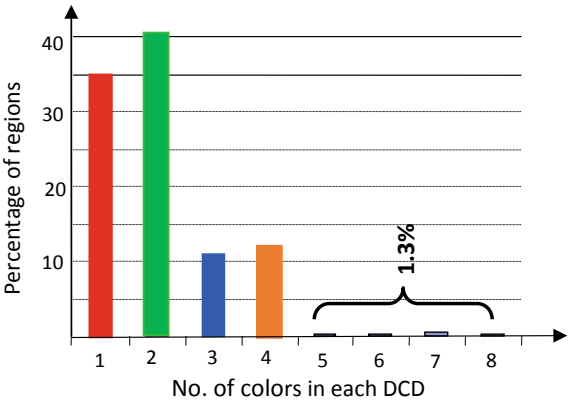

Fig. 4.21 Statistics of DCD numbers from 36,692 regions

The DCD significantly reduces the dimensions of a color histogram, but it still does not address the absence of spatial information from the colors. Therefore, an image is usually segmented into regions and a DCD is extracted from each region of the image.

The number of selected dominant colors in a region depends on the threshold τ. However, statistics based on more than 36,000 image regions show that over 98% of image regions can be described by no more than 4 DCDs (Fig. 4.21) [6–8]. MPEG-7 recommends 1–8 DCDs for each image region.

Figure 4.22 shows some examples of the segmented image regions and their corresponding DCDs [6–8].

Small proportions of colors in a DCD are usually due to segmentation errors or region boundary, they can be discarded without affecting performance. Figure 4.23 shows a few segmented regions and their corresponding DCDs after discarding insignificant colors [6–8].

Region-based DCDs are not only compact but also reflect spatial information in an image, they are a desirable representation for color images. However, unlike conventional color histogram, the order of colors and the number of colors in two DCDs are usually not the same, the matching of two DCDs needs to use many-to-many quadratic matching (12.18).

DCDs can be easily translated into color names, which can be used to describe color images. Human beings tend to describe the visual world using color names, and we can only describe a few hundreds of colors. It is possible to annotate an image with color names based on DCDs [9].

4.4.4 Color Coherence Vector

An ordinary histogram does not tell spatial information about the colors in a bin. The *color coherence vector* (CCV) is a method to incorporate spatial information into a conventional histogram. The idea is to divide each histogram bin into two

Fig. 4.22 Segmented regions and their dominant colors underneath. The dominant colors are shown according to their percentages in the region

Fig. 4.23 Removal of noisy colors from segmented regions. **a** Three sample regions; **b** dominant colors of corresponding regions above; **c** DCDs after discarding insignificant colors from (**b**)

components: coherent (C) and noncoherent (N). The coherent component includes those pixels which are spatially connected, while the noncoherent component includes those pixels that are isolated. A CCV can be computed by using the following procedure:

1. Create a conventional histogram H of k bins for image I using a method in Sect. 4.4.1
2. For each of the histogram bins B_i in H

2.1. Create a binary image I_i from I by marking all pixels with color B_i as 1 (white) and others as 0 (black)
2.2. Set $j = 0$
2.3. For each white pixel p in I_i

2.3.1. If p's West, North West, North, and North East are all black

2.3.1.1. Create a new region R_j
2.3.1.2. $R_j = R_j + p$
2.3.1.3. $j = j + 1$

2.3.2. Otherwise, $R_j = R_j + p$

2.4. For each region R_j

2.4.1. Count the total number of pixels n in R_j
2.4.2. If $n \geq \tau$, $C_i = C_i + n$ //τ is a threshold
2.4.3. Otherwise, $N_i = N_i + n$

3. The normalized sequence $\{(\frac{C_i}{|I|}, \frac{N_i}{|I|}), i = 1, 2, \ldots, k\}$ is the CCV

By dividing each histogram bin into coherent colors and incoherent colors, CCV captures spatial information in an image, it usually performs better than a color histogram. However, the dimension of a CCV is twice of that of a conventional histogram.

4.4.5 Color Correlogram

A color *correlogram* is the color version of *gray level co-occurrence matrix* (Sect. 5.2.2), which is used for texture feature extraction. It characterizes the distribution of color pairs in an image. A color correlogram can be viewed as a 3D histogram, where the first two dimensions represent the colors of any pixel pair and the third dimension is their spatial distance. Thus, in a correlogram, each bin (i, j, k) represents the number of color pairs (i, j) at a distance k.

An input image I is first quantized into m colors $\{c_1, c_2, \ldots, c_m\}$. The computation of a color correlogram is then to find the probability of pixel pairs (p_1, p_2), which meet the following two conditions:

(a) $C(p_1) = c_i$ and $C(p_2) = c_j$
(b) $|p_1 - p_2| = k$

where $C(p_i) = c_i$ means the color of p_i is c_i.
Mathematically, each element of a correlogram is given by

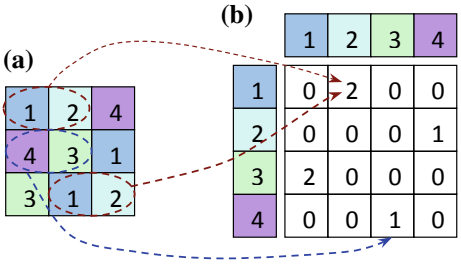

Fig. 4.24 Computation of color correlogram. **a** A 4-color image; **b** the color correlogram of **a** for horizontal distance $k = 1$

$$\gamma_{c_i,c_j}^{k}(I) = Pr\{(p_1, p_2)|C(p_1) = c_i, C(p_2) = c_j \, and \, |p_1 - p_2| = k\} \tag{4.25}$$

If $c_i = c_j$, (4.25) becomes an autocorrelogram which is the probability of finding identical colors at distance k:

$$\alpha_c^k = r_{c,c}^k(I) \tag{4.26}$$

Figure 4.24 shows an example of computing a color correlogram [6]. The color correlogram in Fig. 4.24 is calculated for $k = 1$. Correlograms for other distance $k \in \{1, 2, \ldots, d\}$ can be calculated in similar way.

In total, d correlograms (matrices) can be computed from an image. If the colors are globally quantized, i.e., colors of all images are quantized using a single global palette, two corresponding correlograms from two images can be matched element by element. However, if a local (adaptive) palette is used to quantize each image, either a many-to-many matching is needed or the matching is done through statistics computed from the correlogram matrices (Sect. 5.2.2).

Matching between two color correlograms involves matrix matching which is expensive. In practice, however, only autocorrelograms are used because they are sufficient to produce a good result. Autocorrelograms of each color form a d-dimensional vector, which can be matched using a conventional distance such as L_1 [10].

The performance of the color correlogram is better than the CCV, because it not only captures the special information but also the patterns of the spatial information.

4.4.6 Color Layout Descriptor

It is well known that a spectral transform can capture the frequency of texture changes in an image, and the frequency information is used for identifying most important information from the image. This idea can also be used to capture the frequency of color changes in an image. The *color layout descriptor* (CLD) is just based on this idea.

The computation of CLD consists of four stages: image partitioning, color quantization, DCT transform, and zigzag scanning.

- **Image partitioning**. In the first stage, an input image I is divided into an 8×8 grid of 64 blocks. If the size of the input image is $M \times N$, then the size of each block of the grid will be $(M/8) \times (N/8)$. The reason to divide all images into an equal number of blocks is to ensure resolution or scale invariance.
- **Color quantization**. In the second stage, a single dominant color is computed from each block. The DCD method in Sect. 4.4.2 can be used for the dominant color extraction, but the simplest method is to use the *average* of pixel colors as the representative color. Once the color of each block is quantized, the input image I is converted into an 8×8 color image I_q.
- **DCT transform**. In the third stage, the RGB colors of I_q are converted to Y′CbCr colors (4.21). Then, each of the three Y′CbCr channels of I_q is transformed by an 8×8 discrete cosine transform or DCT, so three sets of 64 DCT coefficients are obtained.
- **Zigzag scanning**. In the final stage, three sets of 64 DCT coefficients are zigzag scanned respectively and the first few coefficients of each set, e.g., 4–8, are chosen. The selected coefficients are then organized into (DY′, DCb, DCr) which is used as the CLD. The reason of only choosing the first few coefficients is because they represent the low-frequency information of the image and they are the most significant coefficients, the remaining coefficients are too small and can be neglected.

CLD allows scalable representation of an image by controlling the number of selected coefficients. MPEG-7 recommends using a total of 12 coefficients, 6 for luminance and 3 for each chrominance, for most of the images. CLD is both compact and scalable, however, it is not robust to rotation change.

4.4.7 Scalable Color Descriptor

As can be seen from the CLD above, a spectral transform like DCT can dramatically reduce the data dimension. This idea can also be used to reduce histogram dimension. A histogram can be regarded as a 1D time series with fluctuations in the vertical direction. Each histogram has a unique pattern of fluctuations or changes along the horizontal direction. This pattern can be effectively captured by using efficient 1D wavelet transform. Because coefficients from a wavelet transform are scalable, i.e., the number of selected coefficients depends on requirement or applications, the result of the wavelet transform is a *scalable color descriptor* or SCD. A SCD is derived using the following procedure.

- **RGB to HSV**. An input image is first converted from RGB color space to HSV color space.

- **Color quantization**. The HSV color space is quantized into 256 colors by dividing the H, S, and V channels into 16, 4, and 4 intervals respectively. A 256-bin color histogram is then created for the image.
- **Bin value quantization**. Each bin value is then nonlinearly quantized into a 4-bit integer to give high significance to small values.
- **Harr wavelet transform**. The histogram is then applied with Harr wavelet transform. Each round of Harr wavelet is a two-pass transform, i.e., low pass and high pass. The low pass of Harr wavelet transform takes two neighboring bins and calculates their sum, while the high pass calculates their difference. Therefore, after the first round, two histograms with half of the original histogram length are obtained: a summed histogram and a differenced histogram. Repeat the transform on the summed histogram for a number of rounds until the two histograms are shortened to the desired length, e.g., 64, 32 or 16 bins.
- **SCD formation**. The final results from the wavelet transform are two short histograms: a summed histogram and a differenced histogram. The two histograms are concatenated to be used as the SCD. However, since the values of the differenced histogram bins are so small that the magnitudes of the bin values are discarded and only the signs of the bins are kept. The sign patterns are sufficient to retain the finer details of the original histogram.

SCD is useful for applications which need short or compact histogram features, however, it does not include spatial information as other color descriptors such as CLD, therefore, its performance is generally lower.

4.4.8 Color Moments

It is well known in data analysis community that descriptive statistics provide a good summary of a dataset and provide a quick understanding of the characteristics or distribution of the dataset such as central tendency, variability, skewness, etc. An image is just a set of color pixel data, therefore, it can also be described by the mean, variance, skewness, etc., which are called *color moments* (CM).

To compute color moments, an input image I is decomposed into individual channels, such as R, G, B channels or H, S, V channels. The moments are then computed from each channel using the following equations:

$$M_1 = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{4.27}$$

$$M_r = \left(\frac{1}{N}\sum_{i=1}^{N}(p_i - M_1)^r\right)^{\frac{1}{r}} \tag{4.28}$$

where

- N is the total number of pixels in image I
- r is the order of a color moment, $r = 2, 3, \ldots$
- p_i is the ith pixel value in the color channel
- M_1 is the first-order color moment, or the mean color of the color channel
- M_2 is the second-order color moment, or the variance of the color channel
- M_3 is the third-order color moment, or the skewness of the color channel

Color moments can also be computed from a color histogram h using the following equations:

$$M_1 = \sum_{k=1}^{K} h_k C_k \tag{4.29}$$

$$M_r = \left(\sum_{k=1}^{K} h_k (C_k - M_1)^r \right)^{\frac{1}{r}} \tag{4.30}$$

where

- h_k is the value of the kth bin of histogram h
- C_k is the color of kth bin of histogram h
- K is the number of bins of h_q
- r is the order of a color moment: 2, 3, …

Typically, only the first three order color moments are computed for a color channel or an image. If three color moments are computed for each color channel, the moments from each of the three channels are concatenated to form a 9-dimensional feature vector which is used to describe the image.

Color moments are a very concise description of an image, however, it can be very inaccurate, e.g., the mean or average color of an image is usually a very coarse description of the image color. Furthermore, color moments do not tell the spatial information of the colors. Therefore, color moments are usually calculated for image regions.

4.5 Summary

Color is often the first feature to be considered during image processing and analysis. A number of preprocessing or preparation are usually performed before the actual color feature extraction such as color space conversion, noise reduction, image scaling, clustering, and segmentation.